(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 447 725 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
08.07.1998 Bulletin 1998/28

(51) Int Cl.[6]: **G06F 15/16**

(21) Application number: **90314393.1**

(22) Date of filing: **28.12.1990**

(54) Updating link state information in networks

Aktualisierung von Verbindungszustandsinformationen in Netzwerken

Mise à jour d'information sur l'état des liaisons dans les réseaux

(84) Designated Contracting States:
DE FR GB IT

(30) Priority: **21.03.1990 US 496632**

(43) Date of publication of application:
25.09.1991 Bulletin 1991/39

(73) Proprietor: **DIGITAL EQUIPMENT CORPORATION**
Maynard, Massachusetts 01754 (US)

(72) Inventors:
- **Perlman, Radia J.**
  Acton, Massachusetts 01720 (US)
- **Kaufman, Charles W.**
  Northboro, Massachusetts 01532 (US)
- **Callon, Ross**
  Bedford, Massachusetts 01730 (US)

(74) Representative: **Goodman, Christopher et al**
Eric Potter Clarkson,
Park View House,
58 The Ropewalk
Nottingham NG1 5DD (GB)

(56) References cited:
- **CONFERENCE PROCEEDINGS ON COMPUTERS AND COMMUNICATION 25 February 1987, SCOTTDALE, AZ, US pages 361 - 367; J.A. DAVIS: 'Integrating communication and database services using intelligent internetwork gateways'**
- **COMPUTER COMMUNICATIONS. vol. 8, no. 6, December 1985, GUILDFORD GB pages 283 - 292; C. SMYTHE: 'Code sequence allocation in a direct sequence spread spectrum local area network'**
- **IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATION. vol. 3, no. 3, May 1985, NEW YORK US pages 416 - 426; A.E. BARATZ: 'SNA networks of small systems'**
- **IEEE TRANSACTIONS ON COMMUNICATION vol. 28, no. 5, May 1980, pages 711 - 719; J.M. MC QUILLAN: 'The new routing algorithm for the ARPANET'**

EP 0 447 725 B1

Description

Background of the Invention

The invention relates to updating link state information in networks.

Information is communicated through such a network along a myriad of links which interconnect nodes. The nodes may be routers, which receive and forward packets toward their destination along the appropriate links. Each router must know the states (e.g., operative or inoperative) of the links in the network in order to send packets along effective paths to their respective destinations, avoiding, for example, faulty links or routers. Schemes for communicating link state information and choosing effective (e.g., optimum) paths are known as routing algorithms.

In popular routing algorithms such as that described in "The New Routing Algorithm for the ARPANET" by McQuillan, Richer, and Rosen, IEEE Transactions on Communications, May, 1980, each node (e.g., router) determines which links are connected to it, the state of those links, and the identity of the node on the other end of each link. To initialize the network, each router places this information in a special packet known as a Link State Packet (LSP), and transmits this LSP to all of the other routers in the network. Later, when changes in the network occur (e.g., a link fails), one or more routers may generate new LSPs which supersede previously generated LSPs. As long as the most recent LSPs are propagated reliably to all of the routers, each router will have complete information about the current topology of the network and can thus use any well known algorithm to compute routes through the network (such as the Dijkstra algorithm described in "A Note on Two Problems in Connexion with Graphs" by Edsgar Dijkstra, Vol. 1, 1959, pages 269-271). However, if the transmission of LSPs becomes unreliable, then the network may eventually become incapable of transmitting data.

When a router receives an LSP, it compares the relative ages of the received LSP and any stored LSPs that were previously received from the same router. The received LSP is stored in place of the stored LSP if the received LSP is more current; otherwise, the stored LSP is retained and the received LSP is ignored. To allow routers to make a determination of the relative ages of LSPs, every LSP is assigned a sequence number; the sequence number of each LSP is greater than that of the LSP that preceded it.

Referring to Fig. 1, a typical LSP includes a source field 10 containing the name of the router that originated the LSP; a sequence number field 12 containing the sequence number for the LSP; and any number of neighbor fields 14, each of which indicates a neighbor node connected to the source router by a communications link.

When changes occur (e.g., the links connected to a router become operative or inoperative), a new version of the router's LSP is issued. In known LSP algorithms, to reflect these changes, fields are removed from or added to the LSP. In known LSP-based routing algorithms, when a link connected to a router fails, the field corresponding to the link (i.e., the field indicating the neighbor to which the link connects) is removed from the router's LSP, and the router's LSP is retransmitted. When a link returns to operation, the field corresponding to the link is added back to the LSP, and the LSP is retransmitted.

In one algorithm for adding and removing fields from a LSP, when a field is removed, each of the fields below the removed field is moved to a position one field nearer to the beginning of the LSP. When a field is added, it is added to the end of the LSP. Thus, when a field near the beginning of the LSP is removed, most of the other fields in the LSP are repositioned.

The LSPs stored by two routers are often compared to determine if the LSPs being stored by one router are more recent than the LSPs being stored by one of its neighboring routers. Sometimes all of the LSPs being stored by two neighboring routers are compared. To accomplish this, one router forms a complete sequence number packet (CSNP) that lists all of the sequence numbers of all of the LSPs currently stored by the router.

Referring to Fig. 2, a CSNP contains the sequence number field 12 of each LSP stored in the database of the router that generated the CSNP. However, these sequence numbers cannot be interpreted without also knowing the identity of the router that generated the sequence numbers. Therefore, the source field 10 of each LSP is paired with the sequence number 12 of the LSP (forming a pair 16).

In a large network, a router may be connected to many communications links, making that router's LSPs very long. Furthermore, there may be many nodes, and thus the CSNPs may also become very large. In general, the longer a packet, the more difficult it is to transmit the packet over a network.

Some network protocols establish a maximum size for packets to be transmitted. For example, on a local area network (LAN) configured in accordance with the IEEE 802.3 protocol (described in the IEEE 802.3 standard, available from the Institute for Electrical and Electronic Engineers, New York, NY), the maximum packet size is approximately 1500 bytes. In order to transmit large packets over such size-limited protocols, a routers must include interface software or hardware. Briefly, using such software or hardware, a sending router splits the large packet into pieces which are smaller than the maximum packet size. These pieces are then transmitted separately to a receiving node. The receiving node then re-assembles the pieces, and the completed large packet is forwarded to a subsequent node (this forwarding may involve re-splitting the packet for transmission over another size-limited protocol).

Even in protocols which are not size-limited, the larger a packet, the more likely that it will be corrupted

by noise when sent along a link. Although communications faults occur at random, the larger a data packet is, the longer it will take to be sent over the link, and thus the more likely it is that a fault will occur during the transmission of the packet. Although it is common for data packets to be encoded with an error-checking algorithm (such as a checksum) which can determine if an error has corrupted the information in the packet, such error-checking algorithms typically do not provide for a way to correct the packet; rather, they can simply determine whether the packet is in error. Therefore, when a packet is received in error, it is discarded by the receiving router. The sending router then re-transmits the packet. The increased probability of errors in transmitting large packets, coupled with the extra time required to transmit such packets, can result in a computation burden on network routers.

Summary of the Invention

In brief summary, the invention as defined in the appended claims features a method for updating, at a first node of a network of nodes interconnected by links, stored information used for routing packets at the first node. The first node receives at least one link state packet indicating the state of one or more links connected to a given node in the network. Thereafter, the first node attempts to derive from the received packet or packets the states of the indicated links. If the states of fewer than all of the links connected to the given node are derived, the stored information is updated using the derived link states. Later, when a link connected to the given node changes from a first state to a second state, if the first state of the changed link was indicated in a previously sent packet, the first node is sent an amended version of the previous packet, which indicates that the changed link is in the second state, and nothing else. An indicator indicating the amendments made to the previous packet is also stored. However, if first state of the changed link was not indicated in a previous packet, the first node is just sent a packet indicating that the changed link is in the second state.

Preferred embodiments include the following features.

The indicator is stored in the given node.

If the changed link subsequently changes from the second state back to the first state, the stored indicator is retrieved, and a twice amended version of the previous packet is formed by removing the amendments indicated by the indicator, and the twice amended version of the packet is sent to the first node.

The first state may be an operative state and the second state an inoperative state, in which case each link state packet indicates that a link connected to the given node is operative by identifying one or more nodes that are connected to the operative link, and indicates that a link connected to the given node is inoperative by omitting identification of one or more nodes that are connected to the inoperative link. In this embodiment, the indicator includes a field for identifying a node and a flag bit for indicating the operability of the link connecting the identified node to the given node.

Other features and advantages of the invention will be appreciated from the following description of the preferred embodiment and from the claims.

Description of the Preferred Embodiment

We first briefly describe the drawings.

Fig. 1 illustrates the contents of a typical LSP.

Fig. 2 illustrates the contents of a typical CSNP.

Figs. 3A through 3C illustrate the contents of LSP fragments.

Fig. 4 is a flow chart of an algorithm for updating the contents of link state packet storage according to the invention.

Figs. 5A through 5C illustrate the contents of a node's link state database.

Fig. 6 is a flow chart of an algorithm for adding neighbors to the fields of the link state database of Figs. 5A through 5C.

Figs. 7A through 7C illustrate the contents of CSNP fragments.

In the invention, rather than sending long LSPs through the network, routers transmit link state information in several smaller fragments. Each fragment is generated, transmitted, and used independently of other fragments. When one of the fragments is corrupted by noise during transmission, only the corrupted fragment is re-transmitted. Furthermore, when the state of a link changes, only (a revised version of) the affected LSP fragment is transmitted.

Note that the "state" of a link, as used below, is information as to whether the link is operative or inoperative; however, a link state fragments formed according to this invention may also include other information about the link such as its "cost". Although described in terms of updating link operability information, the methods described herein are equally applicable to updating these other types of link state information.

A typical set of fragments is illustrated Figs. 3A through 3C. Taken together, the fragments illustrated in Figs. 3A through 3C contain all of the neighbor fields of the prior art LSP of Fig. 1. However, each fragment contains only some of the neighbor fields that are contained in the prior art LSP of Fig. 1. Whereas the LSP in Fig. 1 contains fields for all neighbors (which are numbered 1 through N), the first fragment, shown in Fig. 3A, contains only neighbor fields 1 through K (where $K < N$); the second fragment, shown in Fig. 3B, contains only neighbor fields K+1 through M (where $M < N$); and the third fragment, shown in Fig. 3C, contains only neighbor fields M+1 through N.

Each fragment includes a field 10 indicating the source node; this field contains the same information as the source node field 10 of a prior art LSP (Fig. 1). In

addition, each fragment includes a field 18 indicating the "fragment number", i.e., the fragment's location in the set of fragments that includes all of the link state information of a prior art LSP. Note that the fragment in Fig. 3A is indicated as fragment 1, the fragment in Fig. 3B is indicated as fragment 2, and so on.

Each fragment also includes a field 12 containing a sequence number. This field performs a similar function to the sequence numbers of complete LSPs; however, in this case the sequence number indicates the relative age of the fragment among the versions of the fragment that have been broadcast by the originating router. Because the fragments are independently updated in response to changes in the network, the sequence numbers of various fragments from the same originating node may be different; note that the sequence number of fragment 1 is 12, but the sequence number of fragment 2 is 7, and so on.

Referring to Fig. 4, when a fragment is received by a router, the router searches 13 for any stored fragments having the same source and fragment number as the received fragment. If one is found, the router compares 15 the sequence number of the received fragment to the sequence number of the stored fragment. If the sequence number of the received fragment is higher, or if no stored fragment was found in step 13, the received fragment is stored 17. The router then computes routes through the network based on the link state information included in the received fragment (for example, by computing a routing table using Dijkstra's algorithm). Otherwise, if the stored fragment is more current, the stored fragment is retained and the received fragment is discarded 19.

To enhance the operation of the invention, the relative positions of the neighbor fields of the LSP fragments are carefully maintained. Fig. 5A illustrates the contents of a router's link state database 20 (i.e., the database in the router that stores the states of all of the links connected to the router). The contents and organization of the database 20 correspond to the contents and organization of the fields in the LSP fragments of Figs. 3A through 3C. The first K fields of the database are included in the fragment of Fig. 3A, fields K+1 through M of the database are included in the fragment of Fig. 3B, and fields M+1 through N of the database are included in the fragment of Fig. 3C.

Referring to Fig. 5B, as the states of the links change, the corresponding fields in the link state database and the LSP fragments must change. In Fig. 5B, the links to neighbors M and M+2 have become disabled. In addition, a new link to a neighbor N+1 has become enabled. Because the links to neighbors M and M+2 are no longer operative, subsequent LSP fragments should not indicate these neighbors. Although these neighbors are to be removed from LSP fragments, in the router's database 20, the fields previously occupied by neighbors M and M+2 are not erased or re-used. Rather, the fields for neighbors M and M+2 are marked as "empty" (for example, by setting a flag bit); for future use, the identity of the neighbor that had previously been stored in the field is retained by the field (the identity of the previous occupant is illustrated in parentheses in Fig. 5B). When a new neighbor is added (if there is storage space available in the router's database 20), rather than re-using the fields marked as empty, a new field is added to the end of the database. In Fig. 5B, a new field has been created for new neighbor N+1.

New LSP fragments must be issued to convey the above link state changes to other nodes in the network. These new fragments are illustrated at the bottom of Fig. 5B. Because the field previously occupied by neighbor M is now marked as "empty" in the database 20, the new version of fragment 2 does not have a field for neighbor M, and ends with the field for neighbor M-1. Similarly, because the field previously occupied by neighbor M+2 is now "empty" in the database, the new version of fragment 3 does not have a field for neighbor M+2; the field for neighbor M+1 is followed immediately by the field for neighbor M+3. In addition, because new neighbor N+1 has been added to the database, the new version of fragment 3 ends with a field for the new neighbor N+1. (Alternatively, a new fragment, numbered 4, could be created to contain the field for neighbor N+1).

The states of the links may change again, for example, in Fig. 5C, the link to neighbor M+2 has again become enabled. In this case, because neighbor M+2 is identified as the previous occupant of a field marked "empty", neighbor M+2 is not added to the end of the data base 20. Rather, the field that previously contained neighbor M+2 is modified to not be marked as "empty" (i.e., its flag bit is cleared), and thus neighbor M+2 is again included in the link state database.

In response to the above changes in the link states, a new version of LSP fragment 2 must be issued. This new fragment is illustrated at the bottom of Fig. 5C. Because the field previously marked as "empty" is now occupied by neighbor M+2, the new version of fragment 2 has a field for neighbor M+2; the field for neighbor M+1 is followed immediately by the field for neighbor M+2.

Referring to Fig. 6, in one algorithm, when a neighbor is to be added to the link state database 20, a loop 30, 32, 34 examines the fields in the database which have the empty bit set. Each field is checked 32 to determine if it identifies the neighbor to be added (i.e., if the neighbor to be added had previously occupied the field). If such a field is found, the algorithm clears 36 that field's empty bit, thus re-incorporating the neighbor into the database and the LSP fragments.

If no such field is found, and all of the fields with the empty bit set have been checked (i.e., the answer at step 34 is "yes"), the algorithm checks 38 if the link state database is full. That is, the algorithm checks if there is storage space in the link state database for another field. If the database is not full, a new field is added 40 to the database, and the new neighbor is added to this field. However, if the database is full, then one of the "empty"

fields must be "garbage collected" and used to store the new neighbor. For this purpose, the algorithm selects 42 a field with the empty bit set, and adds the new neighbor to this field. The selection strategy may be random, or, preferably, the selection may be done on a "least recently used" basis. In the latter embodiment, each of the "empty" fields indicates the length of time that they have been marked as empty, and the one that has been marked as empty for the longest time is used to store the new neighbor.

The algorithm of Fig. 6 assumes that the node database has more fields than the total number of links that may be connected to the node. In an alternative embodiment, the algorithm can be modified so that, if this assumption is false, an error is generated. In this embodiment, the algorithm returns an error if, during step 42, a field with the empty bit set is not found.

The above algorithm for adding and removing fields in the node database 20 reduces the extent to which the contents of the neighbor fields move in the database and in the fragments; the removal or addition of a neighbor does not affect the positions of the other neighbors in the database or in the fragments. This feature enhances the benefits of fragmentation: when the state of one link changes, the contents of only one fragment will be affected by the change.

If the addition or subtraction of one link caused the location of many neighbors to change (as is the case in known methods for adding and removing neighbors), the contents of many of the fragments would change. Thus many fragments would have to be updated in response to a change of state of a single link; as a result, transmission overhead would be increased.

The algorithm of Fig. 6 also reduces the possibility that a field for a particular neighbor may move from one LSP fragment to another LSP fragment. Such movements can result in error conditions. For example: if node A's neighbor B moved node A's LSP fragment 2 to node A's LSP fragment from 1, node A would have to broadcast new versions of fragment x and fragment X-1 to the network, because the contents of both fragments changed. If fragment X were broadcast to the network first, for some period of time before fragment X-1 was received by the network, the network would be unaware of the link between A and B, because the old version of fragment X-1 would not indicate the existence of the link, and the new version of fragment X-1 would not yet have been received. This situation may cause an error if a node in the network attempts to route a packet based on the erroneous assumption that there is no link between node A and node B.

As discussed above, it may often be desirable to use CSNPs to compare the contents of the link state databases of two routers, for example, two neighboring routers. Some particular situations where a CSNP comparison may be desirable are as follows: (1) When a link is activated, the link state databases of the two routers that are newly connected by the link should be compared. For this purpose, one or both of the connected routers may transmit CSNPs to the other router. (2) When a new router is first activated (and thus has an empty link state database), the new router's neighbors should provide the new router with their stored LSPs. For this purpose, the new router may transmit an "empty" CSNP to its neighbors, thus requesting an update. (3) On a Local Area Network (LAN), the designated router (which is responsible for routing and "housekeeping" duties for the LAN) should regularly verify that all of the other LAN routers have received recently transmitted LSPs. (This is necessary because routers on a LAN do not send acknowledgement messages acknowledging the receipt of LSPs). For this purpose, the designated router broadcasts a CSNP to the other routers on the LAN.

In large networks, there may be many routers, and thus the network's CSNPs may become very large. The problems that can be caused by large LSPs may also be caused by large CSNPs. Therefore, in an aspect of the invention, large CSNPs are avoided by dispersing the information contained in the CSNP, and transmitting it in several CSNP fragments. These CSNP fragments are illustrated in Figs. 7A through 7C. Each CSNP fragment includes several of the pairs 16 included in the complete CSNP of Fig. 2. However, the order of the pairs 16 in the CSNP fragments is not necessarily the same as the order of the fragments in the complete CSNP.

In CSNPs, the order in which the pairs 16 appear is usually random (and typically depends on the order in which the LSPs were added to the originating router's link state database). In the invention, when creating CSNP fragments, these pairs 16 are ordered. Any suitable ordering scheme (e.g., an alphabetical or numeric scheme) may be used. In a preferred embodiment, the source fields 10 of the pairs 16 are used to determine an ordering of the pairs. Once all of the pairs 16 are ordered, a range of the pairs (e.g., all pairs with source fields numbered 1 through 10) is selected, and those pairs 16 in the selected range are included in a CSNP fragment and transmitted.

For the sake of example, the first CSNP fragment, shown in Fig. 7A, may include the pairs from the sources numbered 1 through X, the second CSNP fragment, shown in Fig. 7B, may include the pairs from sources numbered X+1 through Y, and the third CSNP fragment, shown in Fig. 7C, may include the pairs from the sources numbered Y+1 through Z (where $x < Y < Z$). Note that, although all of the pairs 16 in the selected range are placed in the CSNP fragment, within the CSNP fragment, the pairs do not have to appear in order (i.e., they do not have to be sorted) as shown in Figs. 7A through 7C. If desired, the pairs may be randomly arranged within the CSNP fragment. However, it is preferred to sort the pairs in the CSNP fragment to increase the efficiency with which the CSNP fragment may be parsed and compared to a receiving router's link state database.

To facilitate comparison, two additional fields 50, 52

are included in the header of the CSNP fragments. These fields indicate the range of pairs 16 included in the CSNP fragment. Field 50 indicates the source number (or name) of the numerically (or alphabetically) first pair 16 included in the CSNP fragment. Field 52 indicates the source number (or name) of the numerically (or alphabetically) last pair 16 included in the CSNP fragment. For example, in Fig. 7A, field 50 indicates that source 1 is the start of the range in the CSNP fragment, and field 52 indicates that source X is the end of the range of the CSNP fragment.

The inclusion of fields 50, 52 allows the CSNP fragments to be autonomous, in that their contents are well defined, and can be used independently of the other fragments. This aspect is discussed in more detail below.

When required by the routing algorithm (e.g., under any of the conditions set forth in the background above), a router forms a CSNP fragment, and transmits it to one or more of the other routers in the network. The receiving router first compares the CSNP fragment to the receiving router's link state database to determine whether LSPs for all of the sources mentioned in the CSNP fragment are stored in the receiving router's link state database. This comparison allows the receiving router to determine if there are any LSPs (in the range of the CSNP fragment) stored in the link state database of the router that originated the CSNP, but not stored in the link state database of the receiving router. If there is a LSP missing (i.e., not stored in the receiving router's link state database), the receiving router may, for example, send a packet to the router that originated the CSNP, requesting a copy of the missing LSP.

The receiving router also compares the sequence numbers in the received CSNP fragment to the sequence numbers in the receiving router's link state database. This comparison allows the receiving router to determine the relative age of the LSPs (in the range of the CSNP fragment) stored by the receiving router and by the originating router. If a LSP stored by the receiving router has a higher (i.e., more recent) sequence number than that supplied by the corresponding field 12 of the received CSNP fragment, the receiving router may (possibly depending on the magnitude of the difference between the sequence numbers) send a copy of the more recent LSP to the router which originated the CSNP. Similarly, if a LSP stored by the receiving router has a lower (i.e., less recent) sequence number than that supplied by the corresponding field 12 of the received CSNP fragment, the receiving router may send a packet to the router that originated the CSNP requesting a copy of the more recent LSP.

Finally, the receiving router checks if any other LSPs in its link state database lie in the range specified by the CSNP fragment, but do not appear in the CSNP fragment. This check allows the receiving router to determine if there are any LSPs (in the range of the CSNP fragment) that are stored in the link state database of the receiving router, but are not stored in the link state database of the router that originated the CSNP fragment. If there is a LSP missing (i.e., stored in the receiving router's link state database but not appearing in the CSNP fragment), the receiving router may, for example, send a copy of the missing LSP to the router which originated the CSNP. This last step in particular is made possible by the inclusion of range fields.

In this way, a router receiving a CSNP fragment is able to compare a portion of its link state database to a corresponding portion of the link state database of the router that originated the CSNP fragment. The comparison is as accurate as that provided by known CSNP schemes, but can be made from a single CSNP fragment without regard for the contents of other fragments because the comparison is limited to a range of the LSPs. Therefore, the CSNP fragments do not all have to be received in order to compare the two databases, and, as a result, only some of the CSNP fragments need by transmitted or received at any one time.

A CSNP fragment comparison such as discussed above is limited to a specified range of the two databases, and is thus necessarily incomplete. However, even when a complete CSNP is used to compare two databases, the resulting comparison is only approximate, because the time delays caused by transmission make the CSNP sequence numbers invalid before the CSNP arrives at the receiving node. Thus, database comparisons performed through CSNP fragments (for example, by transmitting a sequence of fragments having ranges which span the entire database) are not necessarily less accurate than comparisons performed with complete CSNPs. A fragment based comparison may in fact be more accurate, because of the reduced transmission overhead (and thus less delay) incurred in transmitting and comparing the smaller fragments.

Other Embodiments

The contents of the CSNP and LSP fragments need not be mutually exclusive; rather, some neighbor fields may be included in more than one fragment.

Other information may be included in LSP fragments. As discussed above, the LSP fragments may include link state information in addition to, or instead of, an indication of whether the link is operable or inoperable. For example, the fragment may include link performance parameters such as "cost". Although described in terms of updating link operability information, the methods described herein are equally applicable to updating these other types of link state information.

Furthermore, LSP or CSNP fragments may include other fields, for example, fields for verifying the contents of the fragments or for aging and invalidating the fragments. In particular, U.S. Patent Application US-A-5 455 865 of Radia J. Perlman filed August 24, 1989 for "Robust Packet Routing over a Distributed Network Containing Malicious Failures", and U.S. Patent Application

US-A-5 086 428 of Radia J. Perlman et. al. filed June 9, 1989 for "Reliable Broadcast of Information in a Wide Area Network", describe additional information which may be included in CSNP or LSP fragments, and algorithms for using this information to enhance the robustness and error recovery of networks.

Claims

1. A method for updating, at a recipient node of a network of nodes interconnected by links, stored information used for routing packets at said first node, characterized by the steps of:

   a. sending one or more link state packets (Fig. 3A) to said recipient node, said packets indicating the states (14) of some but not all of the links connected to a given node in said network,
   b. attempting, at said recipient node, to derive from a link state packet (Fig. 3A) sent in step (a), the states (14) of the links connected to said given node,
   c. updating said stored information used for routing packets using the link states (14) derived in step (b) without regard to other link state packets (Fig. 3A) sent to said recipient node, and
   d. when a given link connected to said given node changes from a first state to a second state:

      i. determining (32) whether the first state of said given link was indicated in a packet (Fig. 3A) previously sent toward said recipient node,
      ii. if the first state of said given link was not indicated in a packet previously sent toward said first node, generating a packet (Fig. 3A) indicating the change of state of said given link, and sending said packet toward said recipient node, otherwise,
      iii. if the first state of said given link was indicated in a packet previously sent toward said first node, generating a replacement for the packet previously sent toward said first node, said replacement packet (Fig. 3A) indicating the change of state of said given link, storing (36) an indicator identifying the change of state indicated by said replacement packet, and sending said replacement packet toward said recipient node.

2. The method of claim 1, wherein said indicator is stored in said given node.

3. The method of claim 1, further comprising

   e. when said given link subsequently changes from said second state back to said first state, retrieving said stored indicator, forming a further replacement packet (Fig. 3A) by removing the changes identified by said indicator, and sending said further replacement packet toward said first node.

4. The method of claim 3, wherein

   in said first state said link is operative, and in said second state said link is inoperative,
   each link state packet indicates that a link connected to said given node is operative by identifying one or more nodes (14) that are connected to the operative link, and indicates that a link connected to said given node is inoperative by omitting identification of any nodes that are connected to the inoperative link, and
   said indicator comprises a field for identifying a node and a flag bit ("EMPTY") for indicating the operability of the link connecting the identified node to said given node.

Patentansprüche

1. Verfahren zum Updaten, bei einem Empfangsknoten eines Netzwerks von durch Verbindungen miteinander verbundenen Knoten, zum Führen von Paketen beim ersten Knoten verwendeter gespeicherter Informationen, gekennzeichnet durch folgende Schritte:

   a. Senden eines oder mehrerer Verbindungszustandspakete (Fig. 3A) zum Empfangsknoten, wobei die Pakete die Zustände (14) einiger, aber nicht aller der Verbindungen anzeigen, die mit einem gegebenen Knoten im Netzwerk verbunden sind,
   b. Versuchen, beim Empfangsknoten, aus einem Verbindungszustandspaket (Fig. 3A), das im Schritt (a) gesendet worden ist, die Zustände (14) der mit den gegebenen Knoten verbundenen Verbindungen abzuleiten,
   c. Updaten der gespeicherten Informationen, die zum Führen von Paketen verwendet werden, unter Verwendung der im Schritt (b) abgeleiteten Verbindungszustände (14) ohne Berücksichtigung anderer zum Empfangsknoten gesendeter Verbindungszustandspakete (Fig. 3A), und
   d. wenn eine gegebene mit dem gegebenen Knoten verbundene Verbindung sich von einem ersten Zustand zu einem zweiten Zustand ändert:

      i. Bestimmen (32), ob der erste Zustand

der gegebenen Verbindung in einem in Richtung zum Empfangsknoten gesendeten Paket (Fig. 3A) angezeigt wurde,
ii. wenn der erste Zustand der gegebenen Verbindung nicht in einem zuvor in Richtung zum ersten Knoten gesendeten Paket angezeigt wurde, Erzeugen eines Pakets (Fig. 3A), das die Änderung eines Zustandes der gegebenen Verbindung anzeigt, und sonst Senden des Pakets in Richtung zum Empfangsknoten,
iii. wenn der erste Zustand der gegebenen Verbindung in einem zuvor in Richtung zum ersten Knoten gesendeten Paket angezeigt wurde, Erzeugen eines Ersatzes für das zuvor in Richtung zum ersten Knoten gesendete Paket, wobei das Ersatzpaket (Fig. 3A) die Änderung eines Zustandes der gegebenen Verbindung anzeigt, Speichern (36) eines Indikators, der die Änderung eines Zustands identifiziert, der durch das Ersatzpaket angezeigt wird, und Senden des Ersatzpakets in Richtung zum Empfangsknoten.

2. Verfahren nach Anspruch 1, wobei der Indikator im gegebenen Knoten gespeichert wird.

3. Verfahren nach Anspruch 1, das weiterhin folgendes aufweist:

e. wenn sich die gegebene Verbindung nachfolgend vom zweiten Zustand zum ersten Zustand ändert, Wiedergewinnen des gespeicherten Indikators, Bilden eines weiteren Ersatzpakets (Fig. 3A) durch Entfernen der durch den Indikator angezeigten Änderungen und Senden des weiteren Ersatzpakets in Richtung zum ersten Knoten.

4. Verfahren nach Anspruch 3, wobei

im ersten Zustand die Verbindung betriebsbereit ist und im zweiten Zustand die Verbindung nicht betriebsbereit ist,
jedes Verbindungszustandspaket anzeigt, daß eine mit dem gegebenen Knoten verbundene Verbindung betriebsbereit ist, durch Identifizieren eines oder mehrerer Knoten (14), die mit der betriebsbereiten Verbindung verbunden sind, und anzeigt, daß eine mit dem gegebenen Knoten verbundene Verbindung nicht betriebsbereit ist, durch Weglassen einer Identifikation irgendwelcher Knoten, die mit der nicht betriebsbereiten Verbindung verbunden sind, und der Indikator ein Feld zum Identifizieren eines Knotens und ein Flag-Bit ("LEER") zum Anzeigen der Betriebsbereitschaft der Verbindung aufweist, die den identifizierten Knoten mit dem gegebenen Knoten verbindet.

Revendications

1. Procédé de mise à jour, au niveau d'un noeud récepteur d'un réseau de noeuds interconnectés par des liaisons, d'informations emmagasinées utilisées pour acheminer des paquets au niveau dudit premier noeud, caractérisé par les étapes consistant:

a. à envoyer un ou plusieurs paquets d'état de liaison (Figure 3A) audit noeud récepteur, lesdits paquets indiquant les états (14) de certaines des liaisons, mais pas de toutes les liaisons, reliées à un noeud donné dudit réseau;
b. à tenter, au niveau dudit noeud récepteur, d'obtenir, d'un paquet d'état de liaison (figure 3A) envoyé à l'étape (a), les états (14) des liaisons reliées audit noeud donné;
c. à mettre à jour lesdites informations emmagasinées utilisées pour l'acheminement de paquets en utilisant les états (14) des liaisons obtenus à l'étape (b) sans tenir compte d'autres paquets d'état de liaison (figure 3A) envoyés audit noeud récepteur; et
d. lorsqu'une liaison donnée reliée audit noeud donné passe d'un premier état à un second état:

i. à déterminer (32) si le premier état de ladite liaison donnée était indiqué dans un paquet (figure 3A) envoyé précédemment vers ledit noeud récepteur;
ii. si le premier état de ladite liaison donnée n'était pas indiqué dans un paquet envoyé précédemment vers ledit premier noeud, à générer un paquet (figure 3A) indiquant le changement d'état de ladite liaison donnée, et à envoyer ledit paquet vers ledit noeud récepteur ; sinon,
iii. si le premier état de ladite liaison donnée était indiqué dans un paquet envoyé précédemment vers ledit premier noeud, à générer un remplacement pour le paquet envoyé précédemment vers ledit premier noeud, ledit paquet de remplacement (figure 3a) indiquant le changement d'état de ladite liaison donnée, à emmagasiner (36) un indicateur identifiant le changement d'état indiqué par ledit paquet de remplacement, et à envoyer ledit paquet de remplacement vers ledit noeud récepteur.

2. Procédé selon la revendication 1, dans lequel ledit

indicateur est emmagasiné dans ledit noeud donné.

3. Procédé selon la revendication 1, consistant, en outre:

e. lorsque ladite liaison donnée repasse subséquemment dudit second état audit premier état, à récupérer ledit indicateur emmagasiné, à former un autre paquet de remplacement (3A) en supprimant les modifications identifiées par ledit indicateur, et à envoyer ledit autre paquet de remplacement vers ledit premier noeud.

4. Procédé selon la revendication 3, dans lequel:

dans ledit premier état, ladite liaison est en fonctionnement et, dans ledit second état, ladite liaison est hors fonctionnement;
chaque paquet d'état de liaison indique qu'une liaison reliée audit noeud donné est en fonctionnement en identifiant une ou plusieurs noeuds (14) qui sont reliés à la liaison en fonctionnement, et indique qu'une liaison reliée audit noeud donné est hors fonctionnement en omettant l'identification de tous les noeuds qui seraient reliés à ladite liaison hors fonctionnement; et
ledit indicateur comporte une zone pour identifier un noeud et un bit de drapeau ("EMPTY") pour indiquer l'état de fonctionnement de la liaison reliant le noeud identifié audit noeud donné.

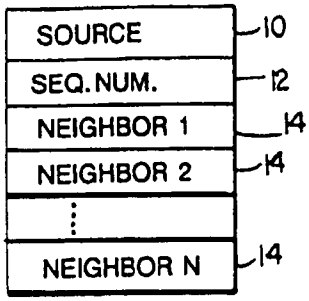


FIG.1
PRIOR ART

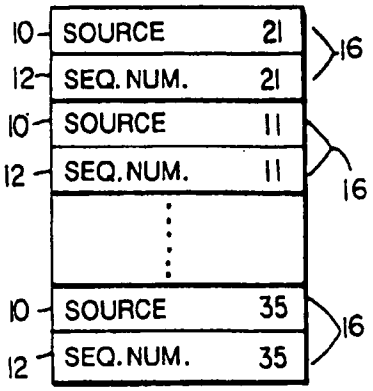


FIG.2
PRIOR ART

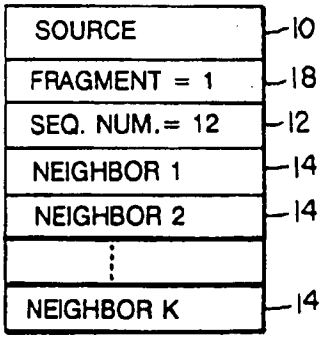


FIG.3a

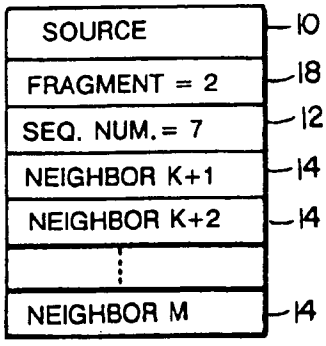


FIG.3b

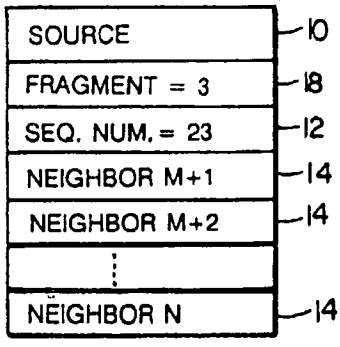


FIG.3c

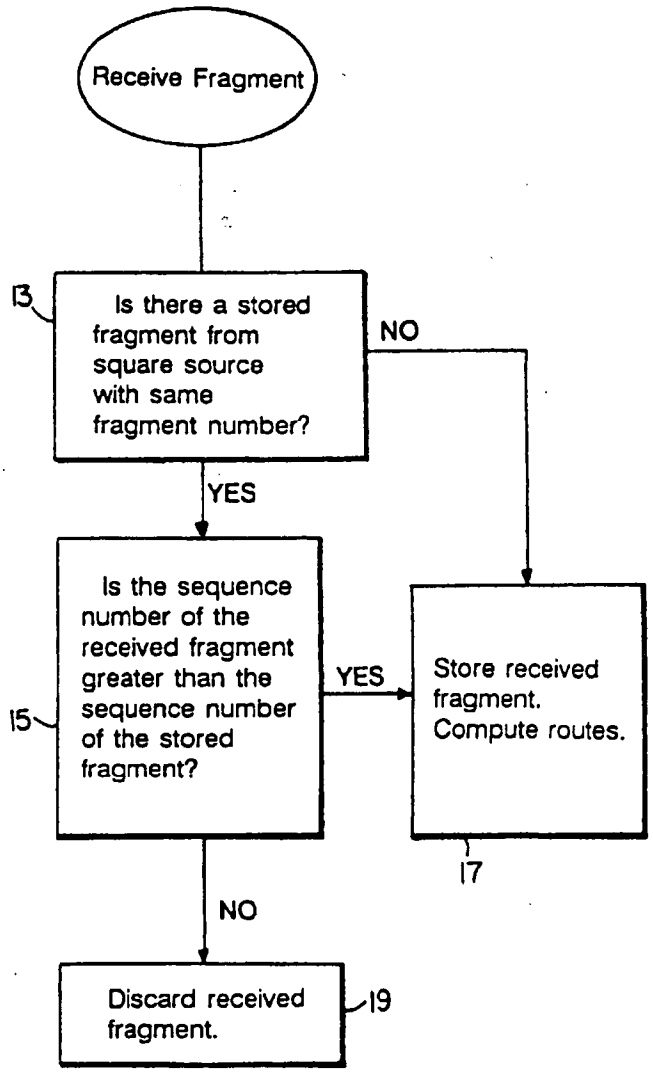

Receive Fragment
13
Is there a stored fragment from square source with same fragment number?
NO
YES
Is the sequence number of the received fragment greater than the sequence number of the stored fragment?
15
YES
Store received fragment. Compute routes.
17
NO
Discard received fragment.
19

FIG.4

FIG.5a

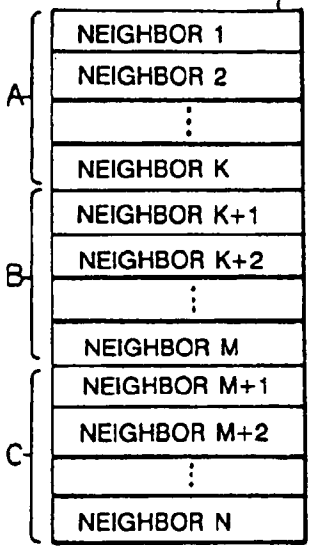


FIG.5b

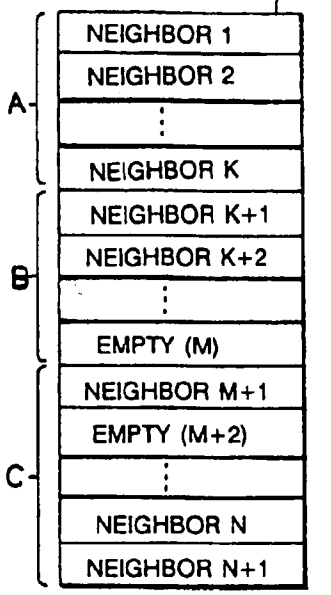
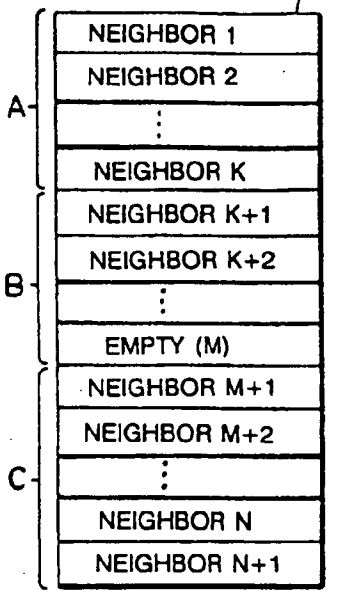
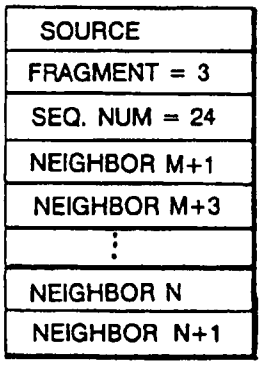
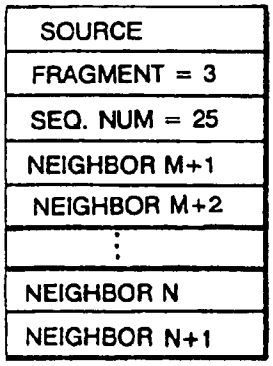
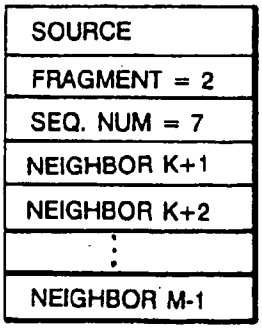


FIG.5c









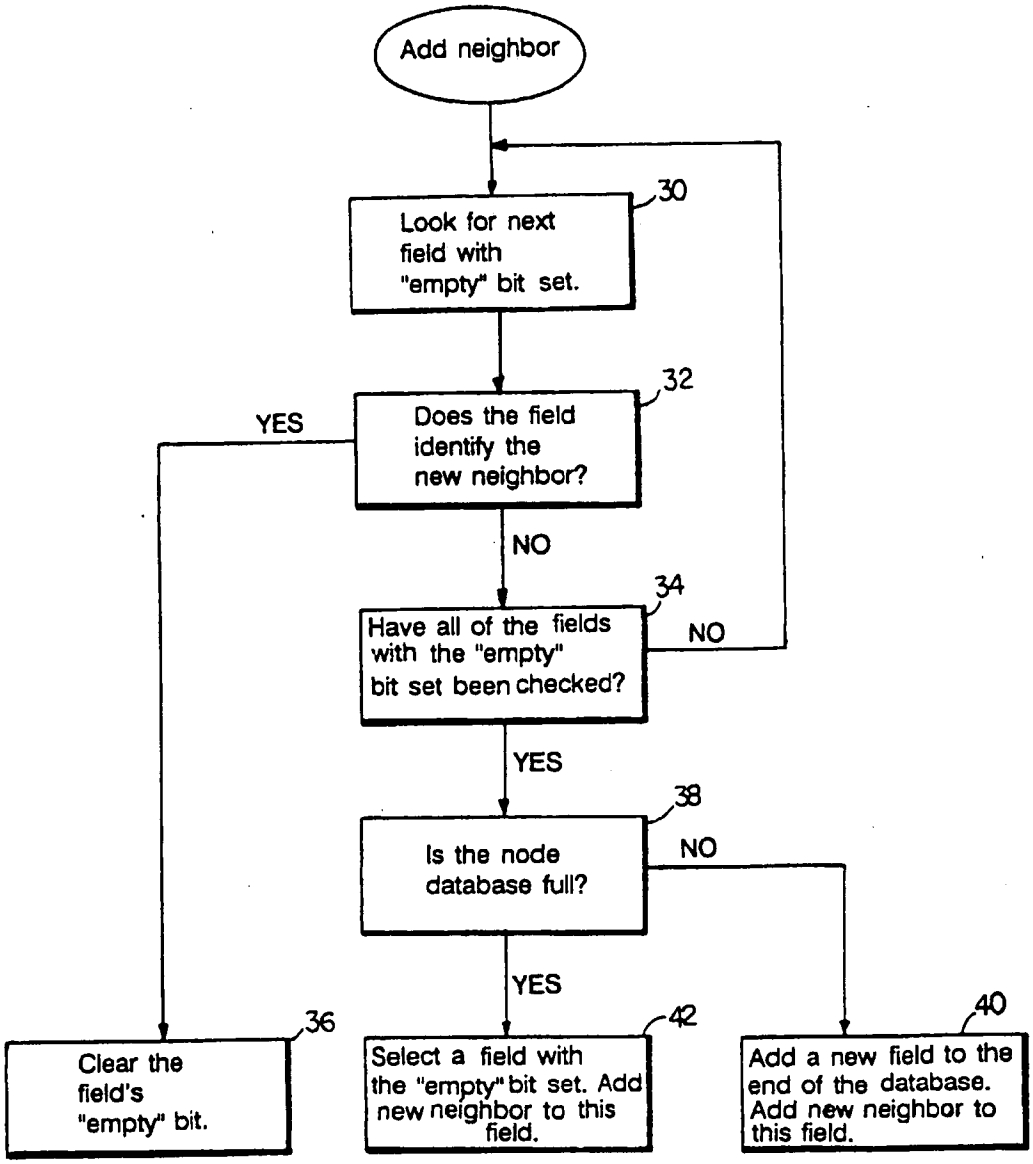

Add neighbor
Look for next field with "empty" bit set.
30
Does the field identify the new neighbor?
32
YES
NO
Have all of the fields with the "empty" bit set been checked?
34
NO
YES
Is the node database full?
38
NO
YES
Clear the field's "empty" bit.
36
Select a field with the "empty" bit set. Add new neighbor to this field.
42
Add a new field to the end of the database. Add new neighbor to this field.
40

FIG.6

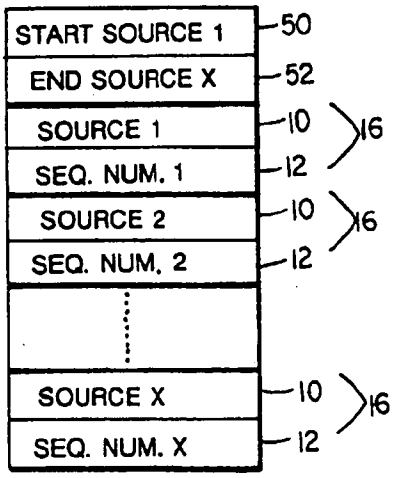


FIG. 7a

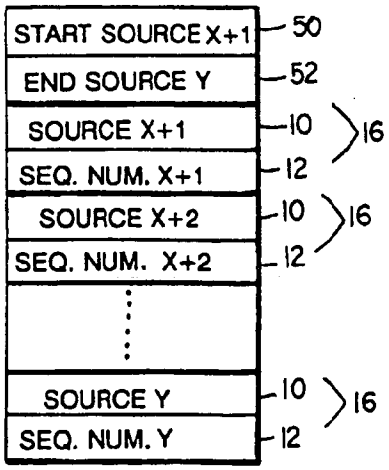


FIG. 7b

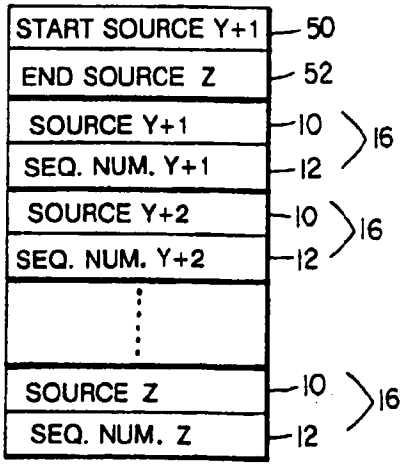


FIG. 7c